Schedule Code Generator for SubSystem

Given an affine system with subsystems/useEquations, this page shows how to specify the targetmapping for the system and generate the code.

Matrix Multiplication with subsystem

The following code is the alpha program for matrix matrix multiplication with dot-product subsystem.

```
affine matrix_product_SubSyst {N,K,M | N>0 && K>0 && M > 0}    // Product
between a N*K matrix and a K*M matrix
input
    float A {i,k | 0<=i<N && 0<=k<K};
    float B {k,j | 0<=k<K && 0<=j<M};
output
    float C {i,j | 0<=i<N && 0<=j<M};
let
    use {iP,jP|0<=iP<N && 0<=jP<M} dot_product[K]
((pi,pj,k->pi,k)@A,(pi,pj,k->k,pj)@B) returns (C);
.

affine dot_product {N| N>0}  // Product between 2 vector of size N
input
    float vect1 {i | 0<=i<N };
    float vect2 {i | 0<=i<N };
output
    float Res;
local
    float temp {i | 0<=i<N};
let
    temp[i] = case
        {i|i==0}   : vect1[0] * vect2[0];
        {i |0<i<N} : temp[i-1] + vect1[i]*vect2[i];
    esac;
    Res[] = temp[N-1];
.
```

The program contains two systems. The dot_product system takes two vectors as inputs and computes the doc product of these two vectors. The matrix_product_SubSyst computes matrix C=A*B, the (ip,jp)th element for the answer matrix C is computed by calling the dot product subsystem, and the (ip)th row of A, and (jp)th column of B is passed as input to the subsystem call.

TargetMapping for the Matrix Multiplication Example

The schedule code generator treats every subsystem call (an instance of the subsystem) as an function call in C. In order to ensure the correctness of the code, each instance of the subsystem call is attached with three special statement by default. The three statements are memory allocation statement, value copy statement and memory free statement. The memory allocation statement allocates a temporary variable for the corresponding input, and the value copy statement copies the corresponding values into the temporary variable before it is passed into the function call, and the memory free statement frees the memory when the temproary is not useful.

In order for schedule code generator to generate the code, other than specify the schedule for the useEquation, a schedule has to be specified for each special statement of each input/ouput of the useEquation.

The following command set the SpaceTimeMap for the (n)th input/output of a useEuqation identified with lable:

```
setSpaceTimeMapForUseEquationOptimization(program, system, label, isInput,
n, SpaceTimeMapForMalloc, SpaceTimeMapForValueCopy,
SpaceTimeMapForMemoryFree);
```

The parameter isInput sepcifies whether the SpaceTimeMap is specified for input or not, and the last three prarameter specifies the space time map for the three special statement attached to the current input/output. Those specification can also be specified separately with the following command:

```
setSpaceTimeMapForMemoryAllocation(program, system, label, isInput, n,
SpaceTimeMap);
setSpaceTimeMapForValueCopy(program, system, label, isInput, n,
SpaceTimeMap);
setSpaceTimeMapForMemoryFree(program, system, label, isInput, n,
SpaceTimeMap);
```

The following script consists all the commands that specifies the TargetMapping for the matrix multiplication example and generates the code.

```
prog = ReadAlphabets("./Matrix_product_SubSyst.ab");
rootSystem = "matrix_product_SubSyst";
subSystem = "dot_product";
outDir = "./test-out/"+ rootSystem;
CheckProgram(prog);

#set the spacetimeMap for the subSystem first
setSpaceTimeMap(prog, subSystem, "temp", "(i->i)");
setSpaceTimeMap(prog, subSystem, "Res", "(->N)");

#the identification lable for the useEquation
label = "UseEquation_C";

#set the spacetimeMap for the rootSystem
```

```
#set the spacetimeMap for the main subsystem call
setSpaceTimeMap(prog, rootSystem, label, "(ip,jp->1,ip,jp,2)");

#set the SpaceTimeMap for the first input of the useEquation
setSpaceTimeMapForMemoryAllocation(prog, rootSystem, label, 0, 0,
"(ip,jp->0,0,0,0)");
setSpaceTimeMapForValueCopy(prog, rootSystem, label, 0, 0,
"(ip,jp->1,ip,jp,0)");
setSpaceTimeMapForMemoryFree(prog, rootSystem, label, 0, 0,
"(ip,jp->2,0,0,0)");

#set the SpaceTimeMap for the second input of the useEquation
setSpaceTimeMapForUseEquationOptimization(prog, rootSystem, label, 0, 1,
"(ip,jp->0,0,0,1)", "(ip,jp->1,ip,jp,1)", "(ip,jp->2,0,0,1)");

#set the spaceTimeMap for the first output of the useEquation
setSpaceTimeMapForUseEquationOptimization(prog, rootSystem, label, 1, 0,
"(ip,jp->0,0,0,2)", "(ip,jp->1,ip,jp,3)", "(ip,jp->2,0,0,2)");

generateScheduledCode(prog, rootSystem, outDir);
generateWrapper(prog, rootSystem, outDir);
generateMakefile(prog, rootSystem, outDir);
```

TargetMapping for Optimization

The schedule code generator generates three special statements for each input/output of each useEquation. However, those special statements can be saved under some situation. For example, the <iP, jP>th value of the final matrix C is computed by a dot product of the <iP>th row of matrix A and the <jP>th column of matrix B by the <iP, jP>th instance of the use equation. Assume that the memory for matrix A, B and C are all allocated in row-wise major, the <iP>th row of matrix A can be passed as a pointer of A, no temporary variable is needed. This can be achieved by specifying the memory space for the first input of the useEquation to be the same as A.

The following command is the command that specifies the optimization for the first input of the useEquation C. <sxh cs; gutter:false> setMemorySpaceForUseEuqationOptimization(prog, rootSystem, label, 0, 0, “A”); <\sxh>

From:
https://www.cs.colostate.edu/AlphaZ/wiki/ - **AlphaZ**

Permanent link:
https://www.cs.colostate.edu/AlphaZ/wiki/doku.php?id=schedule_code_generator_for_code_with_subsystem&rev=1404769066

Last update: **2014/07/07 15:37**